Image Processing and Computer Vision

Luca Domeniconi

June 17, 2024

Disclaimer

This document contains the answer for some question for the exam of Image Processing and Computer Vision. Don't use them as a replacement for the study and for the professor slides.

Acknowledgements

A lot of answers are inspired by Tian Cheng "Riccardo" Xia notes whom I really thanks. Moreover, thanks to Nicolas Cridlig for the review.

Module 1

Question 1

Describe the shape-based matching method. Provide a list of the main steps, with a brief description for each step. Then, describe the similarity function used for the matching with the relative mathematical formulation and the motivation behind his use. Finally, specify the variants of the similarity function that can be exploited to overcome issues that may arise in shape-based matching.

Shape-based matching is an edge-based matching technique to find a template image T inside a scene image I and is composed of the following steps:

1. Use an edge detector to extract a set of control points $\{P_1, \dots, P_n\}$ from the template image T (not from the scene image I).

2. Compute the gradient normalized at each point P_k:

$$\nabla T(P_k) = \begin{pmatrix} \partial_x T(P_k) \\ \partial_y T(P_k) \end{pmatrix} \qquad u_k(P_k) = \frac{\nabla T(P_k)}{\|\nabla T(P_k)\|}$$

3. Given a patch $\tilde{I}_{i,j}$, defined as a crop of the scene image of the same dimensions of the template with the upper left corner in position (i, j), compute the gradient normalized at the points $\{\tilde{P}_1(i,j), \dots, \tilde{P}_n(i,j)\}$

$$\nabla \tilde{I}_{i,j}(\tilde{P}_k) = \begin{pmatrix} \partial_x \tilde{I}_{i,j}(\tilde{P}_k) \\ \partial_y \tilde{I}_{i,j}(\tilde{P}_k) \end{pmatrix} \qquad \tilde{u}_k(\tilde{P}_k) = \frac{\nabla \tilde{I}_{i,j}(\tilde{P}_k)}{\|\nabla \tilde{I}_{i,j}(\tilde{P}_k)\|}$$

   Note: $u_k(P_k) \cdot \tilde{u}_k(\tilde{P}_k) = \|u_k(P_k)\| \cdot \|\tilde{u}_k(\tilde{P}_k)\| \cdot \cos(\theta) = 1 \cdot 1 \cdot \cos(\theta) = \cos(\theta)$, where θ is the angle between the two vectors.

4. Use a similarity function to see if the gradients match. Different similarity function has been proposed:

   - **Cosine similarity** (can be seen as a mean of the cosines of the angles between each pair $(u_k, \tilde{u}_k)$):

$$S(i,j) = \frac{1}{n} \sum_{k=1}^{n} u_k(P_k) \cdot \tilde{u}_k(\tilde{P}_k) = \frac{1}{n} \sum_{k=1}^{n} \cos(\theta_k) \in [-1, 1]$$

     If $S(i,j) = 1$ it means that all the vectors representing the control points perfectly match. In practice, a minimum threshold $S_{\min}$ is used to determine if there is a match.

   - **Global polarity inversion** if the object is on a lighter background or darker background, the sign of all the cosines is inverted and in order to solve an absolute value of the sum can be used:

$$S(i,j) = \frac{1}{n} \left| \sum_{k=1}^{n} u_k(P_k) \cdot \tilde{u}_k(\tilde{P}_k) \right| = \frac{1}{n} \left| \sum_{k=1}^{n} \cos(\theta_k) \right| \in [0, 1]$$

- **Local polarity inversion** if parts of the object are on a lighter background or a darker background, the sign of the cosines will not match and in order to solve this, an absolute value on the single cosine is used:

$$S(i,j) = \frac{1}{n}\sum_{k=1}^{n}\left|u_k(P_k)\cdot\tilde{u}_k(\tilde{P}_k)\right| = \frac{1}{n}\sum_{k=1}^{n}|\cos(\theta_k)| \in [0,1]$$

Question 2

Describe the bilateral filter, its main objective and how it works. In this respect, briefly highlight its benefits compared to linear filters. Then, describe the relative mathematical formulation, detailing each component involved. Finally, provide an analysis of its limitations.

The bilateral filter is a non-linear filter whose goal is to deal with Gaussian noise without the introduction of blur. It is basically a Gaussian filter that is able to ignore pixels belonging to different objects, leaving the edges clear. It is able to do that using two different Gaussians, one taking into account the spatial distance of two pixels and the other the intensity distance. In this way, if two pixels are near in the image but have very different intensities (they probably belong to different objects), they won't be considered when calculating the resulting pixel value.

On the other hand, linear filters apply the same kernel everywhere and so are not able to distinguish between different objects or understand if they're near an edge.

Given two pixels positions p and q and an image I, the following can be computed:

- **Spatial distance**: $d_s(p,q) = \|p-q\|_2$
- **Range/Intensity distance** $d_r(p,q) = |I(p)-I(q)|$

Given a pixel p, its neighborhood $\mathcal{N}(p)$ and the variances σ_s, σ_r of the two Gaussians, the bilateral filter applied on p is computed as follows:

$$O(p) = \sum_{q\in\mathcal{N}(p)} H(p,q)\cdot I(q)$$

where $H(p,q)$ is defined as:

$$H(p,q) = \frac{G_{\sigma_s}(d_s(p,q))G_{\sigma_r}(d_r(p,q))}{\sum_{z\in\mathcal{N}(p)} G_{\sigma_s}(d_s(p,z))G_{\sigma_r}(d_r(p,z))}$$

The main limitation of the bilateral filter is its computational complexity. Being a non-linear filter, its kernel changes for every pixel and this leads to slow performance. Another limitation might be the choice of the parameters σ_s and σ_r, that can lead to manual tuning and may not generalize well across different images.

Question 3

List the main steps of the DoG keypoint detection algorithm, providing a very brief description for each step. Report the equation to define sigma for each scale in an octave. Describe the motivations behind exploiting multiple octaves and how they are computed.

Difference of Gaussians (DoG) keypoints detection algorithm is composed of the following steps:

1. **Gaussian Scale-Space Construction**: compute a series of images by applying Gaussian filtering with progressively increasing σ to the input image, creating a scale-space representation. The σ used for the i-th image in the scale space is defined as $\sigma_i = k^i \cdot \sigma_0$ for $i = 0, 1, 2, \ldots, s$, where $k = 2^{1/s}$, and s is the number of intervals per octave.

2. **Difference of Gaussians (DoG) Computation**: generate DoG images by subtracting adjacent Gaussian-blurred images in the scale-space. Each DoG image highlights the edges and blobs at a particular scale. It is defined as:

$$\begin{aligned} \texttt{DoG}(x, y, \sigma) &= \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned}$$

3. **Extrema Detection in Scale-Space**: identify keypoints by finding local extrema (maxima or minima) in the DoG images and doing non-maxima suppresion. Each pixel in a DoG image is compared with its 26 neighbors (8 in the same scale and 9 in the scales above and below). If it's the maximum in the neighborhood, it is defined as a keypoint.

4. **Keypoint Filtering**: points with a weak DoG response can be pruned through thresholding. Furthermore, it has been observed that strong DoG points along edges are unstable and can also be pruned.

Using multiple octaves allows the algorithm to detect keypoints that are invariant to scale changes. This means keypoints can be consistently detected even when the image undergoes a significant scaling transformation. Multiple octaves are computed by repeatedly downsampling the image by a factor of 2 and then applying the scale-space process to each downsampled image. This ensures that keypoints of various sizes are detected efficiently across a wide range of scales.

Question 4

(A) Describe how SIFT descriptors are computed and (B) how they can be matched between a target view and a reference (or model) view. In this respect, (C) highlight how potentially wrong matches may be filtered out.

(A) The SIFT Descriptors are computed in the following steps:

1. **Detection of Keypoints**: The Scale-Invariant Feature Transform (SIFT) begins with detecting keypoints using a Difference of Gaussian (DoG) method. This involves:

   - Constructing a scale-space by convolving the input image with Gaussian filters at different scales.
   - Subtracting adjacent Gaussian-blurred images to obtain DoG images.
   - Identifying local extrema (both minima and maxima) in the DoG images by comparing each pixel with its neighbors in the current and adjacent scales.

2. **Descriptor Computation**: for every found keypoint the SIFT descriptor is computed by:

    - Centering a 16×16 grid on the keypoint, which is divided into 4×4 subregions.
    - For each subregion, an orientation histogram with 8 bins (45° each) is computed, resulting in a $4 \cdot 4 \cdot 8 = 128$-dimensional feature vector.
    - The gradients are weighted by a Gaussian function to reduce the influence of gradients far from the keypoint center.
    - The descriptor is normalized to unit length to enhance invariance to illumination changes.

The SIFT descriptor is thus invariant to scale, rotation, and affine intensity changes.
(B) To match SIFT descriptors between a target view and a reference view a Nearest-Neighbor Search is done. This typically involves computing the Euclidean distance between the SIFT descriptors of keypoints.

To determine if a match is valid, the following criteria are used:

- **Threshold**: the match is accepted if the distance to the nearest neighbor is below a predefined threshold.
- **Ratio of Distances**: the match is accepted if the ratio of the distance to the nearest neighbor and the distance to the second nearest neighbors is below a threshold (usually 0.8). This helps in reducing ambiguity when two potential matches are close to each other.

(C) To filter out wrong matches, two main technique are used:

- **Ratio test**: the ratio test described above with a threshold of 0.8 is empirically found to reject 90% of incorrect matches while only missing 5% of good matches.
- **RANSAC**: using RANSAC it's possible to compute the homography from the target view to the reference view ignoring possible outliers (bad matches).

Module 2

Question 5

List, without describing in detail, the main steps of the zhang's camera calibration algorithm. Then, discuss how many different intrinsic and extrinsic parameters must be estimated if we collect 12 calibration images. Finally, describe how the intrinsic parameters A and extrinsic parameters are computed once homographies for each calibration image have been estimated.

These are the main steps of Zhang's Camera Calibration algorithm:

1. **Image acquisition**: Capture multiple images of a planar pattern (e.g., a chessboard).
2. **Initial Homographies Guess**: Compute an initial guess of the homography for each image.
3. **Non-linear Refinement of Homographies**: Refine the homographies obtained through a non-linear minimization process.
4. **Initial Intrinsic Parameters Guess**: Estimate the intrinsic camera parameters.
5. **Initial Extrinsic Parameters Guess**: Estimate the extrinsic parameters for each image.
6. **Initial Distortion Parameters Guess**: Estimate the distortion parameters.
7. **Non-linear Optimization**: Refine all the parameters through a global optimization process.

For 12 calibration images:

- **Intrinsic parameters**: 5
- **Extrinsic parameters**: 6 for each image (3 for rotation, 3 for translation):
  - Total extrinsic parameters: $12 \times 6 = 72$
- **Total parameters**: $5 + 72 = 77$

Computation of Intrinsic and Extrinsic Parameters from Homographies

NOT COMPLETE

Question 6

Discuss the limitations of fully connected linear layers when used to process $H \times W$ grayscale images. Then discuss the inductive biases of convolutions and how they help to overcome the previous limitations. Finally, report the expression to computes each entry of the output activation, clearly defining all the terms used in it.

Fully connected (FC) linear layers are inherently limited when processing $H \times W$ grayscale images due to several reasons:

- An FC layer treats the input image as a single vector of size $H \times W$. For an image of size $H \times W$, an FC layer requires $(H \times W) \times (H \times W - 1)$ parameters and $2(H \times W) \times (H \times W - 1)$ FLOPs, which scales quadratically with the image size. This leads to a massive number of parameters and high computational cost.

- FC layers do not preserve the spatial structure of the image. Each pixel is treated independently, ignoring the local relationships and spatial hierarchies present in images.

CNNs address these limitations through specific inductive biases:

- Convolutional layers connect each neuron to only a local region of the input image. This local receptive field allows the model to capture spatial hierarchies and local features, such as edges and textures.

- In convolutional layers, the same filter (set of weights) is applied across different regions of the image. This parameter sharing significantly reduce the number of parameters compared to FC layers. For an image of size $H \times w$, a convolution requires only a fixed number of parameters (equal to the size of the filter).

- Convolutional layers maintain translation equivariance, meaning that shifting the input image results in a correspondingly shifted output feature map. This property is crucial for tasks where the spatial location of features is not important, such as image classification.

For a convolutional layer, the output activation at each position (j, i) is computed using the following expression:

$$[K * I](j,i) = \sum_{n=1}^{C_{in}} \sum_{m=-\lfloor \frac{H_K}{2} \rfloor}^{\lfloor \frac{H_K}{2} \rfloor} \sum_{l=-\lfloor \frac{W_K}{2} \rfloor}^{\lfloor \frac{W_K}{2} \rfloor} K_n(m,l) I_n(j-m, i-l) + b$$

where:

- K is the convolution kernel (filter).

- I is the input image.

- C_{in} is the number of input channels.

- H_K and W_K are the height and the width of the kernel, respectively.

- $K_n(m,l)$ represents the value of the kernel at position (m,l) for the n-th input channel.

- $I_n(j-m, i-l)$ represents the value of the input image at position $(j-m, i-l)$ for the n-th input channel.

- b is the bias term associated with the filter.

Question 7

Describe grouped convolutions with G groups, discuss how many kernels of which shape they use to realize a $C_{out} \times C_{in} \times 3 \times 3$ convolution and compare them to standard convolutions in terms of number of parameters and FLOPS, motivating your answer. Then, discuss how grouped convolutions are used to realize depthwise separable convolutions.
Not explained by professor: *Finally, draw the inverted residual block and the squeeze and excitation ResNet block, highlighting if, where and why they used grouped convolutions.*

Grouped convolutions divide the input channels into G groups, where each group is convolved separately. This means that instead of having a single convolutional operation over all input channels, we have G smaller convolutions. For a convolution with C_{out} output channels and C_{in} input channels using 3×3 kernels, the grouped convolution uses G groups, each handling C_{in}/G input channels and producing C_{out}/G output channels.

If we denote the kernel size as 3×3, each group has this many parameters:

$$\frac{C_{out}}{G} \times \frac{C_{in}}{G} \times 3 \times 3$$

Therefore, the total number of parameters for G groups is:

$$G \times \left(\frac{C_{out}}{G} \times \frac{C_{in}}{G} \times 3 \times 3 \right) = C_{out} \times \frac{C_{in}}{G} \times 3 \times 3$$

Compared to standard convolutions that uses $C_{out} \times C_{in} \times 3 \times 3$ parameters, grouped convolution divides the parameter count by G. FLOPs are reduced similarly. Standard convolution requires $H \times W \times C_{out} \times C_{in} \times 3 \times 3$, whereas grouped convolutions only require $H \times W \times C_{out} \times \frac{C_{in}}{G} \times 3 \times 3$

Grouped convolutions are then used to realize depthwise separable convolutions, which consist of two stages:

- **Depthwise Convolution**: Each input channel is convolved separately with its own filter, using C_{in} convolutions of 3×3 each.
- **Pointwise Convolution**: A 1×1 convolution is applied to combine the outputs of the depthwise convolution, using $C_{out} \times C_{in} \times 1 \times 1$ kernels.

This reduces the computational cost significantly compared to standard convolutions.

Question 8

Describe in detail the motivating experiment for the introduction of Residual Networks (ResNet) and discuss if it shows underfitting or overfitting, justifying your answer and explaining why the architecture of ResNets should reduce it. Then, draw the two variants of residual blocks used in ResNets and compute their numbers of parameters and FLOPs. Use them to justify the introduction of the bottleneck variant. Finally, discuss why and how the standard residual block is modified at the beginning of a stage.

Motivating Experiment for the Introduction of ResNet

The introduction of ResNet was motivated by the observation that training very deep neural network was not leading to better performance. Specifically, the experiment revealed that as the depth of a network increased, its training error increased, which was counterintuitive because deeper networks should, in theory, capture more complex features and thus perform better.

The experiment showed that even when deeper networks had the capacity to model the training data, they performed worse than shallower networks, not due to overfitting, but underfitting. This was because the optimization algorithm (gradient descent) struggled to train very deep networks effectively. Residual Networks addressed these issues by introducing shortcut connections (or skip connections), which allowed gradients to flow more easily through the network, thereby mitigating the vanishing gradient problem and enabling the training of much deeper networks.

Overfitting vs. Underfitting

The motivating experiment demonstrated underfitting rather than overfitting. Overfitting occurs when a model performs well on training data but poorly on test data, usually because it has learned noise and irrelevant details from the training data. In contrast, underfitting occurs when a model is too simple to capture the underlying pattern of the data, resulting in a poor performance on both training and test data. The inability of very deep networks to perform well on training data indicated that they were underfitting, as they could not be optimized effectively due to the vanishing gradient problem.

Architecture of ResNet and Its Benefits

ResNets mitigate underfitting by using residual blocks with shortcut connections. These connections allow the network to learn the identity mapping more easily, which ensure that deeper networks perform at least as well as shallower ones. The architecture of ResNet reduces the optimization difficulty by allowing the gradients to bypass certain layers, which helps in maintaining a stronger gradient flow throughout the network.

Residual Blocks in ResNet

The standard residual block consists of:

- Two 3×3 convolutional layers, each followed by batch normalization and ReLU activation.
- A skip connection that adds the input of the block to its output.

Note that the second ReLU of the residual block is placed after the addition of the input. Otherwise the output of each block could only get bigger and bigger, as ReLU always outputs a number ≥ 0.

The bottleneck residual block is designed to increase the depth of the network while leaving the computational cost unchanged. It consists of:

- A 1×1 convolution to reduce the number of channels.
- A 3×3 convolution.

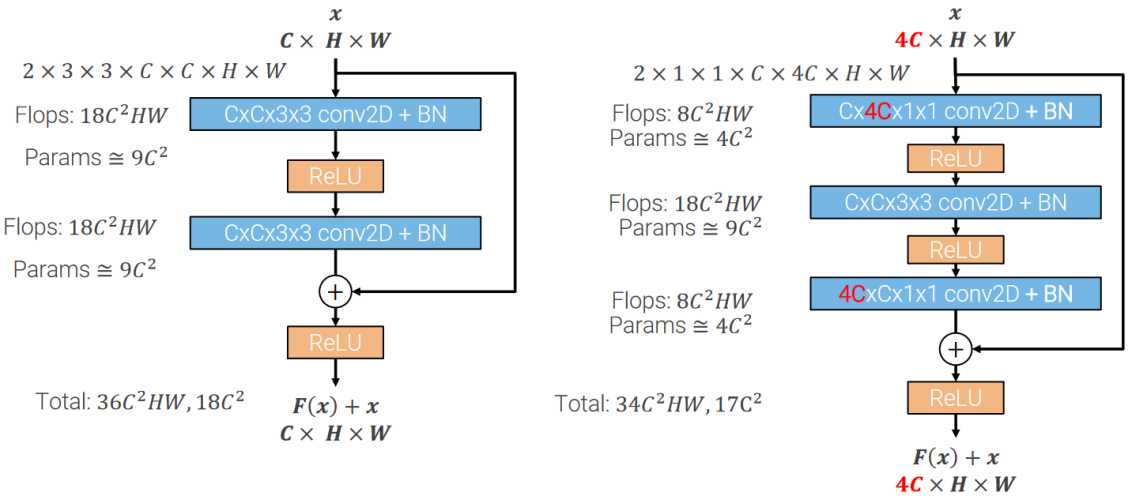

Figure 1: Standard residual block (left) and bottleneck block (right)

- Another 1×1 convolution to restore the number of channels.
- A skip connection that adds the input of the block to its output.

Number of parameters and FLOPs

By convention the parameters are indicated in this order $H_{kernel} \times W_{kernel} \times C_{out} \times C_{in}$

- **Standard Residual Block**: for an input with C channels and spatial dimensions $H \times W$
  - Each 3×3 convolution has $3 \times 3 \times C \times C = 9C^2$ parameters (ignoring the biases).
  - There are two such convolutions, so total parameters $= 2 \times 9C^2 = 18C^2$.
  - FLOPs are similarly calculated: each 3×3 convolution over $H \times W$ image involves $2 \times (9C^2) \times H \times W = 18C^2 \times H \times W$, so total FLOPs per block $= 36C^2 \times H \times W$
- **Bottleneck Residual Block**: for an input with C channels and spatial dimensions $H \times W$
  - The first 1×1 convolution has $1 \times 1 \times C \times 4C = 4C^2$ parameters (ignoring biases).
  - The 3×3 convolution has $3 \times 3 \times C \times C = 9C^2$ parameters.
  - The second 1×1 convolution has $1 \times 1 \times 4C \times C = 4C^2$
  - Total parameters $= 4C^2 + 9C^2 + 4C^2 = 17C^2$
  - For the FLOPs, the first 1×1 convolution involves $2 \times (4C^2) \times H \times W = 8C^2 \times H \times W$ operations.
  - The 3×3 convolution involves $2 \times (9C^2) \times H \times W = 18C^2 \times H \times W$ operations.
  - the second 1×1 convolution involves $2 \times (4C^2) \times H \times W = 8C^2 \times H \times W$ operations (same as first 1×1 convolution).
  - Total FLOPs $= (8C^2 + 18C^2 + 8C^2) \times H \times W = 34C^2 \times H \times W$

Modifications at the Beginning of a Stage

At the beginning of each stage in a ResNet, the spatial dimension is typically halved, and the number of channels is doubled. Leaving the skip connection unchanged would cause an error during the sum because of the different shapes of the two tensors. To fix the shape of the skip connection, a 1×1 convolution with stride 2 and $C_{out} = 2C_{in}$ is usually applied, in order to halve the spatial dimension and double the number of channels.